MULTER

NESTJS

UPLOAD FILE

www.sfectoria.com

Multer

Introduction

في أي App نستحقو باش نعملو upload لتصاور ، فيديوات و حتي documents, مالمهم أنو نفهمو كيفية تخزين ال files هاذم في ال server side

ال flow من ال front لل back في العادة شفنا إنو الback ينجم يستقبل الtype : الapplication-json , Number , Text .

توا باش نشوفو content -type جديد متاع ال file إلي هي 'multipart/form-data' فهل ال back ينجم يستقبلو ؟

Multer

What is Multer

- ال multer هو middleware ي handle تحميل الملفات كيما تصاور ، فيديوات ... الmulter فيه configuration باش تحط ال uploaded files في dossier في ال server
- ف ال réponse هي إنو من أشهر les librairies إلي تخلينا نستقبلو ال 'multipart/form-data' في ال server side هي ال multer .
- باش ن upload ال files في ال server متاعنا لازمنا 2 مراحل

Multer

1st Handling File Uploads in NestJS with FileInterceptor

```ts
import { Controller, Get, Post, UploadedFile, UseInterceptors } from '@nestjs/common';
import { FileInterceptor } from '@nestjs/platform-express';
import { Multer } from 'multer';
```

```ts
@Post('upload')
 @UseInterceptors(FileInterceptor('file'))

 uploadFile(@UploadedFile() file: Multer.File) {
   console.log(file,"file");
 }
```

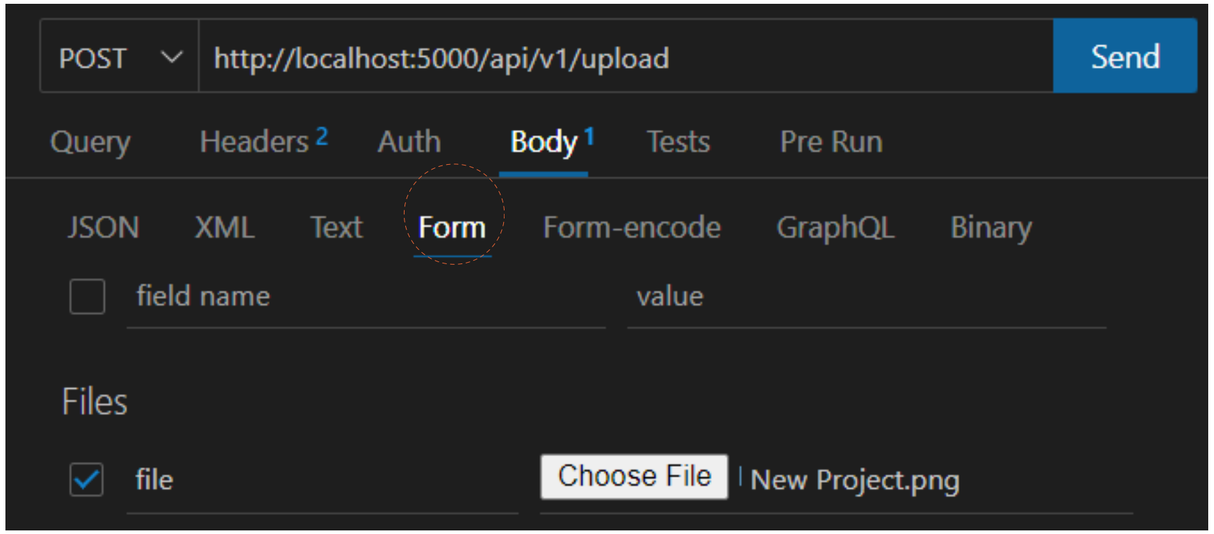
ال interceptors في ال nestjs يخليوك تنجم تعمل حاجات قبل ما ال request توصل لل controller ولا بعد ما تخرج مل controller

ال file نحب نعمل عليه modifications قبل ما يوصل لل request

ال decorator هذا @uploadFile() باش يقول لل nestjs باش ي extract ال uploaded file من ال incoming request و ت bindiha مع ال file

Multer

1st Test it with Thunder




```
} {
  fieldname: 'file',
  originalname: 'New Project.png',
  encoding: '7bit',
  mimetype: 'image/png',
  buffer: <Buffer 89 50 4e 47 0d 0a 1a 0a 00 00 00
 0d 49 48 44 52 00 00 00 c1 00 00 00 bf 08 06 00 0
0 00 47 6a b8 2d 00 00 00 01 73 52 47 42 01 d9 c9
2c 7f 00 00 00 09 ... 49543 more bytes>,
  size: 49593
}
```

- ال console.log(file) باش تعطينا informations عل uploaded file منغير ما تسجلو
- أما المشكلة هوني في ال buffer إلي هو ال binary data متاع ال uploaded file خاترو رزين برشا
- كان باش ن savih كيما هو باش يعملي slow performance و increase database size كيف نsavih في ال database

Multer

2nd Advanced File Handling & Storage with Multer in NestJS

- فالحل إني ن Stori الfile في dossier و بعد وقت حاشتي بيها جيست ن accedilha بالpath متاعها

```
1  import { existsSync, mkdirSync } from 'fs';
2  import { extname } from 'path';
3  import { Multer, diskStorage } from 'multer';
```

```
1  const multerConfig = {
2    dest: 'upload',
3  };
```

- أول حاجة ن import ال packages إلي باش نستحقوهم و ن declare multerConfig فيه إسم dossier إلي باش يتحطو فيه التصاور

```
@UseInterceptors(
    FileInterceptor('file', {
      storage: diskStorage({
        destination: (req: any, file: any, cb: any) => {
          const uploadPath = multerConfig.dest;
          //  Create folder if doesn't exist
          if (!existsSync(uploadPath)) {
            mkdirSync(uploadPath);
            console.log('mkdir upload');
          }
          cb(null, uploadPath);
        },
        filename: (req: any, file: any, cb: any) => {
          //  Generating a 32 random chars long string
          const randomName = Array(32)
            .fill(null)
            .map(() => Math.round(Math.random() * 16).toString(16))
            .join('');
          //  Calling the callback passing the random name generated with
          // the original extension name
          cb(null, `${randomName}${extname(file.originalname)}`);
        },
      }),
    }),
  )
```

باش نزيدو لل FileInterceptor في ال parameter متاعو objet باش يكون فيه configuration options باش ن handle ال file storage

- ال diskStorage نستعملوه باش ن store ال uploaded file في ال disk متاع ال server متاعك
- ال destination function باش تحدد فيها وين باش تsave ال uploaded file في ال disk

2nd Explaining the previous code

the second parameter of the FileInterceptor (configuration options object)

Multer

Generating a Random String to the filename:

- **Array(32).fill(null)** creates an array with 32 elements, all initialized with null.
- **.map(() => Math.round(Math.random() * 16).toString(16))** maps over each element of the array and generates a random hexadecimal character (0-9, a-f) for each element. This is achieved by:

    - **Math.random() * 16** generates a random number between 0 and 15.
    - **Math.round(...)** rounds the random number to the nearest whole number, ensuring it falls within the range of 0-15.
    -**.toString(16)** converts the rounded number to its hexadecimal representation (e.g., 10 becomes 'a', 15 becomes 'f').
    -**.join('')** concatenates all the random hexadecimal characters together into a single string.

Appending Original File's Extension:

- **`${randomName}${extname(file.originalname)}`** combines the randomly generated string (**randomName**) with the original file's extension (**extname(file.originalname)**).
- **extname(file.originalname)** extracts the extension of the original filename. For example, if the original filename is 'example.jpg', **extname(file.originalname)** would return '.jpg'.
- **${randomName}${extname(file.originalname)}** combines the random string and the original file extension to create a unique filename.

Multer

Example:

- Suppose the uploaded file is named 'example.jpg'.
- The random string generation may produce something like 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6'.
- The original file's extension is '.jpg'.
- Combining these, the generated filename would be 'a1b2c3d4e5f6g7h8i9j0k1l2m3n4o5p6.jpg'.

Multer

The process generates unique filenames by combining a random string of hexadecimal characters with the original file's extension. This prevents filename conflicts when multiple files are uploaded.

the console.log(file,”file”) become :

```
} {
  fieldname: 'file',
  originalname: 'New Project.png',
  encoding: '7bit',
  mimetype: 'image/png',
  destination: 'upload',
  filename: 'd0fa2bca666e2e68fe3b67bc3c51e224.png',
  path: 'upload\\d0fa2bca666e2e68fe3b67bc3c51e224.png',
  size: 49593
}
```

Multer

you have successfully uploaded
the file to the server!

Congrats!